TÓM LƯỢC BÀI GIẢNG

(Vũ Quốc Hoàng)

CON TRỎ VÀ CẤP PHÁT ĐỘNG

Chủ đề

- Con trỏ và cấp phát động

Tài liệu

[1] Tony Gaddis, *Starting out with C++ From Control Structures through Objects*, Pearson, 8th edition, 2015.

[2] Vũ Quốc Hoàng, *Bí kíp luyện Lập trình C (Quyển 1)*, hBook, 2017.

Đọc tài liệu

- Cấp phát bộ nhớ động: Mục 9.8 [1]

Kiến thức

- Đoạn mã và hình minh họa 1 (lưu ý là các địa chỉ cụ thể 0x62d0, … chỉ có tính minh họa)

```
#include <stdio.h>
#include <stdlib.h>

void input(double a[], int n)
{
    for(int i = 0; i < n; i++)
    {
        printf("Nhap so thu %d: ", i + 1);
        scanf("%lf", &a[i]);
    }
}

double mean(double *a, int n)
{
    double s = 0;
    for(int i = 0; i < n; i++)
        s += *(a + i);
    return s/n;
}

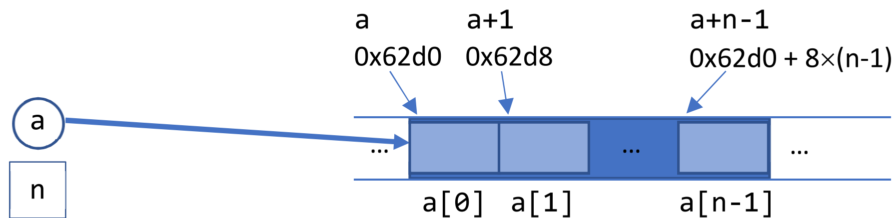
int main()
{
    double *a;
    int n;
```

```
    printf("Ban muon lam viec voi bao nhieu so? ");
    scanf("%d", &n);

    a = (double*)malloc(n * sizeof(double)); // a = new double[n];
    if(a == NULL)
        printf("Khong cap phat bo nho duoc!!!");
    else
    {
        input(a, n);
        printf("Trung binh cac so la: %.2lf", mean(a, n));

        free(a); // delete[] a;
    }
}
```

Vùng nhớ cục bộ (Stack) **Vùng nhớ cấp phát động (Heap)**



- Khi biết trước số lượng dữ liệu cần dùng trong chương trình và số lượng này ít thì ta nên dùng các biến để chứa dữ liệu.
- Khi lượng dữ liệu nhiều nhưng ít biến động và bị chặn trên thì ta có thể dùng *mảng tĩnh* (static array) để chứa dữ liệu. Mảng tĩnh là mảng được cấp phát trước trên *vùng nhớ cục bộ* (Stack) với kích thước cố định, dự phòng theo lượng chứa tối đa. Việc dùng mảng tĩnh có nhược điểm là lãng phí nếu không dùng hết và không thể chứa quá sức chứa định trước. Các mảng một chiều, hai chiều đã học là các mảng tĩnh.
- Khi lượng dữ liệu biến động nhiều theo yêu cầu *lúc chạy* (runtime), khó xác định chặn trên và để tránh lãng phí thì ta có thể dùng *mảng động* (dynamic array). Mảng động là mảng được *cấp phát động* (dynamically allocate) trên *vùng nhớ cấp phát động* (Heap) với kích thước tùy ý theo yêu cầu. Con trỏ được dùng để quản lý các vùng nhớ cấp phát động vì các vùng nhớ này không đặt tên được. Để cấp phát động ta dùng hàm `malloc` (`calloc`, `realloc`) của thư viện chuẩn C trong file tiêu đề `stdlib.h`. Trong C++ ta có thể dùng lệnh cấp phát động với toán tử new.
- Việc cấp phát động có thể không thực hiện được khi không còn vùng nhớ trống. Lúc đó các hàm/toán tử cấp phát động sẽ trả về con trỏ NULL. Đây là địa chỉ đặc biệt, địa chỉ 0, thường được dùng với ý nghĩa "địa chỉ không hợp lệ" hay "không trỏ". NULL thực chất là hằng tượng trưng được `#define` là `0` trong thư viện chuẩn C.
- Vùng nhớ cấp phát động sau khi dùng xong cần được *giải phóng* (deallocate). Trong C ta dùng hàm `free` (cũng trong file tiêu đề `stdlib.h`). Trong C++ ta dùng lệnh giải phóng với

toán tử `delete`. Nhớ rằng: <u>nên</u> giải phóng vùng nhớ đã cấp phát động khi không cần dùng nữa nhưng <u>không được</u> giải phóng vùng nhớ không được cấp phát động trước đó hay vùng nhớ đã được giải phóng.

- Đoạn mã và hình minh họa 2 (lưu ý là các địa chỉ cụ thể 0x62d0, ... chỉ có tính minh họa)

```
#include <stdio.h>
#include <stdlib.h>

void fill_pascal(int *a[], int n)
{
    for(int i = 0; i < n; i++)
        a[i][0] = a[i][i] = 1;
    for(int i = 1; i < n; i++)
        for(int j = 1; j < i; j++)
            a[i][j] = a[i-1][j-1] + a[i-1][j];
}

void output_pascal(int **a, int n)
{
    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j <= i; j++)
            printf("%-5d", *(*(a + i) + j));
        printf("\n");
    }
}

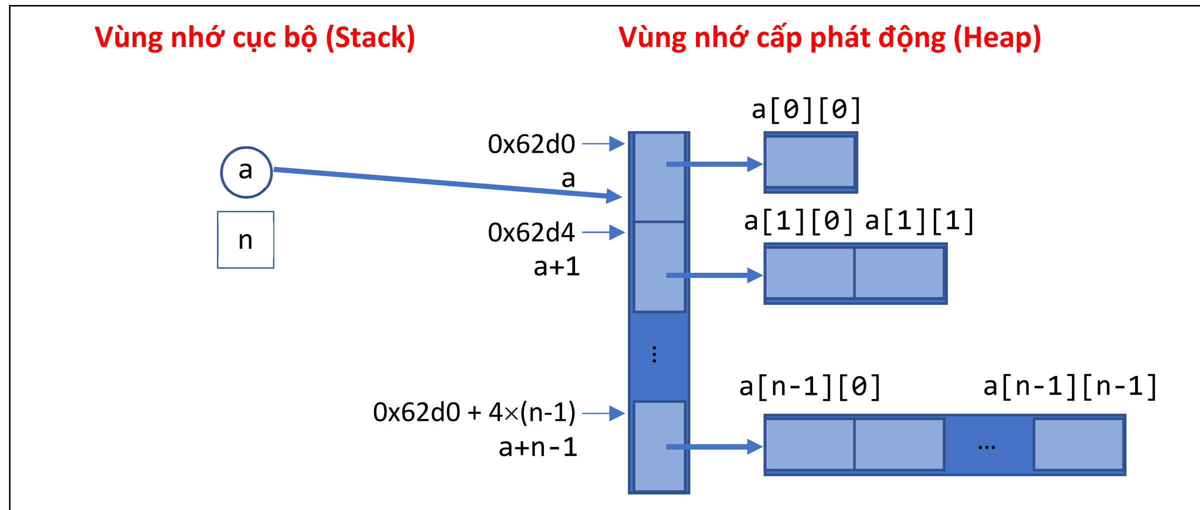
int main()
{
    int **a;
    int n;

    printf("Nhap so dong: ");
    scanf("%d", &n);

    a = new int*[n]; // a = (int**)malloc(n * sizeof(int*));
    for(int i = 0; i < n; i++)
        a[i] = new int[i + 1]; // a[i] = (int*)malloc((i+1) *
sizeof(int));

    fill_pascal(a, n);
    output_pascal(a, n);

    for(int i = 0; i < n; i++)
        delete[] a[i]; // free(a[i]);
    delete[] a; // free(a);
}
```



- Đoạn mã trên minh họa kĩ thuật cấp phát động mảng 2 chiều, đúng hơn là *mảng tam giác* (triangular array). Cần nắm rõ về con trỏ để có thể hiểu và vận dụng được kĩ thuật này. Đặc biệt kĩ thuật này dùng *con trỏ cấp hai*, tức là con trỏ trỏ tới con trỏ.
- (Nâng cao: Khác biệt giữa mảng 2 chiều tĩnh và mảng 2 chiều cấp phát động.)
- (Nâng cao: Mô phỏng mảng nhiều chiều trên một vùng nhớ liên tục (như mảng một chiều).)

Kĩ năng

- Thành thạo các kí pháp tương đương giữa mảng và con trỏ
- Thành thạo kĩ thuật cấp phát động mảng một chiều và mảng hai chiều
- Vận dụng cấp phát động và con trỏ để tổ chức và xử lý dữ liệu

Lưu ý

- Cấp phát động là kĩ thuật mạnh mẽ và quan trọng trong việc tổ chức và xử lý dữ liệu nên sinh viên cần nắm vững

Bài tập

Làm lại Bài tập 4.1.1, 4.1.2, 4.2.4, 4.2.8 [2] bằng kĩ thuật cấp phát động